Neural MoCon: Neural Motion Control for Physically Plausible Human Motion Capture —Supplementary Material—

Buzhen Huang Liang Pan Yuan Yang Jingyi Ju Yangang Wang

Southeast University, China

In the supplementary material, we first introduce implementation details of our method for reproducing the experimental results (Sec. 1). An additional evaluation on different metric is provided (Sec. 2). Then, more results on different datasets are shown to demonstrate the performance of our method (Sec. 3). We further discuss the results in the supplementary video (https://www.bilibili.com/video/BV1W94y1f7ht) (Sec. 4). Finally, the superiority of the proposed neural motion control is discussed (Sec. 5).

1. Implementation details

We adopt PyBullet [1] as simulator. The control frequency is 240HZ, and the coefficient of friction is 0.9. Since the frame rate among videos is different, we apply linear interpolation on the estimated motion between two frames to obtain reference pose and velocity. The frequency of sampling is 30HZ. To train the distribution prior, we implement the neural network based on PyTorch [10]. The distribution encoder and the pose decoder have six and four fully-connected layers, respectively, with batch normalization and LeakyReLU [9] activation function. The AdamW [8] optimizer with a learning rate of 0.0001 is used to train the network. On a desktop with an Intel(R) Core(TM) i9-11900F CPU and a GPU of NVIDIA GeForce RTX 3090, one sample takes about 0.0002s without any implementation acceleration strategy. We sample 1000 samples for a target pose and save 20 samples as the start state for the next target pose.

1.1. Physical character creation

In this section, we explain the details of physical character creation with different body shape variations. To represent the kinematic and dynamical model in a unified framework, we design the kinematic tree of the physical character to be the same as the SMPL [7]. According to the estimated SMPL shape parameters, we automatically generate a new character. With the joint regressor in SMPL, we obtain the length of each bone of the estimated SMPL model in T-pose. Since the bones in symmetrical parts have minor difference, we calculate the average length and correct the rotation for each bone, and build the skeleton based on parent-child relationship. Further, we determine the link shape with the created skeleton. The physical characters with different shapes are shown in Fig. 1. In addition, we do not control the hand and foot, so that these joints are fixed. Since the control parameters are dramatically affected by mass, all characters in our experiments have the same mass. The details of the character model are described in Tab. 1.

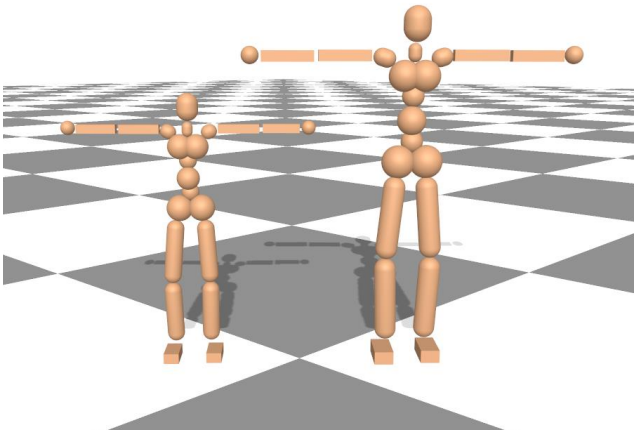

Figure 1. The physical character with different body shapes. The character has the same joint positions as its corresponding SMPL model.

1.2. CMA-ES

The CMA-ES (covariance matrix adaptation evolution strategy) [2] is a black-box optimization method. We implement this algorithm with [3] to prepare pseudo ground-truth for distribution prior training. The mean and the variance have the same dimension as target pose, which is 51. The number of maximum resampling is 100 and the population size is 6 in our experiments. The distribution evolves 30 generations for a given character state and reference pose. To get more natural motion, we limit the sampling bounds, which is shown in Tab. 3.

Joint	Type	Geometry	Mass	Num	Kp	Kd	Force Limit	Inertia(xx)	Inertia(xy)	Inertia(xz)	Inertia(yy)	Inertia(yz)	Inertia(zz)
Lower Neck	revolute	capsule	0.5	1	200	20	100	0.001	0.0	0.0	0.001	0.0	0.001
Upper Neck	revolute	capsule	3.0	1	200	20	100	0.001	0.0	0.0	0.001	0.0	0.001
Chest	revolute	sphere	8.0	1	500	50	300	0.001	0.0	0.0	0.001	0.0	0.001
Lower Back	revolute	sphere	5.0	1	500	50	300	0.001	0.0	0.0	0.001	0.0	0.001
Upper Back	revolute	sphere	5.0	1	500	50	300	0.001	0.0	0.0	0.001	0.0	0.001
Clavicle	revolute	capsule	1.0	2	400	40	200	0.001	0.0	0.0	0.001	0.0	0.001
Shoulder	revolute	box	2.0	2	400	40	200	0.001	0.0	0.0	0.001	0.0	0.001
Elbow	revolute	box	1.0	2	300	30	150	0.001	0.0	0.0	0.001	0.0	0.001
Wrist	fixed	sphere	0.5	2	–	–	–	0.001	0.0	0.0	0.001	0.0	0.001
Hip	revolute	capsule	5.0	2	500	50	300	0.001	0.0	0.0	0.001	0.0	0.001
Knee	revolute	capsule	3.0	2	400	40	200	0.001	0.0	0.0	0.001	0.0	0.001
Ankle	revolute	box	1.0	2	300	30	100	0.001	0.0	0.0	0.001	0.0	0.001

Table 1. Control parameters and joint information. Inertia (ij) represents the link inertia coefficient between the i-axis and j-axis.

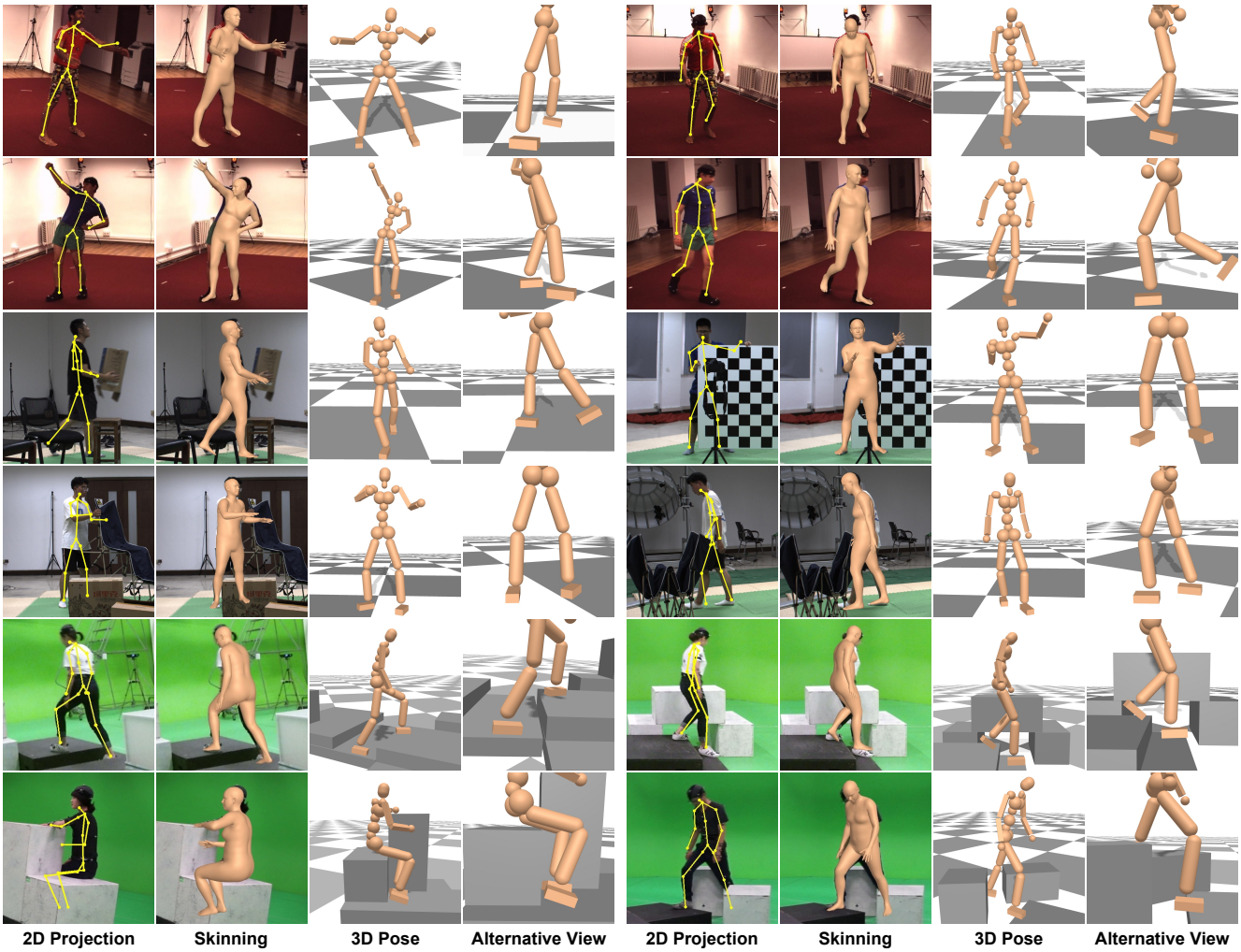

Figure 2. Qualitative results on Human3.6M (row 1-2), 3DOH (row 3-4) and GPA (row 5-6) dataset. Our method is robust to complex terrains, occlusions and body shape variations. We can obtain natural skinning mesh with the estimated pose parameters.

1.3. Training details

We introduce the training details of our distribution prior in this section. As mentioned in the main paper, the train set from Human3.6M and GPA are used for training. We first apply the CMA-ES method to get pseudo ground-truth. Since generating sampled target pose for a complete motion sequence is difficult and time-consuming, we select two consecutive frames from the dataset and calculate the state of character from the dataset with linear interpolation. The kinematic pose in the second frame is used as reference. We then apply CMA-ES method to obtain the target pose distribution. When the prior is convergent, we finish the pre-train procedure and incorporate the two-branch decoder to refine the network. It is an ideal situation to have the character

Character Property	Value	Simulator Property	Value
Joints	19	Gravity	9.81
Movable Joints	17	Time Step	1/240.0
Fixed Joints	2	NumSolverIterations	10
Links	19	NumSubSteps	2
Total Mass (kg)	53.5		
Degrees of Freedom	57		
Lateral Friction Coefficient	0.9		
Rolling Friction Coefficient	0.3		
Restitution Coefficient	0.0		

Table 2. Properties of the physical character and the physics simulator.

Joint	-x	+x	-y	+y	-z	+z
Left Hip	-2.0	2.0	-0.57	0.57	-0.27	0.27
Left Knee	-0.3	1.57	-0.27	0.27	-0.0	0.0
Left Ankle	-0.57	0.57	-0.57	1.2	-0.57	0.57
Right Hip	-2.0	2.0	-0.57	0.57	-0.27	0.27
Right Knee	-0.3	1.57	-0.27	0.27	-0.0	0.0
Right Ankle	-0.57	0.57	-1.2	0.57	-0.57	0.57
Lower Back	-1.57	1.57	-1.57	1.57	-1.57	1.57
Upper Back	-1.57	1.57	-1.57	1.57	-1.57	1.57
Chest	-1.57	1.57	-1.57	1.57	-1.57	1.57
Lower Neck	-0.57	0.57	0.0	0.0	0.0	0.0
Upper Neck	-0.57	0.57	-0.57	0.57	0.0	0.0
Left Clavicle	-1.57	1.57	-1.57	1.57	-1.57	1.57
Left Shoulder	-1.57	1.57	-1.57	1.57	-1.57	1.57
Left Elbow	-1.57	1.57	-1.57	1.57	-1.57	1.57
Right Clavicle	-1.57	1.57	-1.57	1.57	-1.57	1.57
Right Shoulder	-1.57	1.57	-1.57	1.57	-1.57	1.57
Right Elbow	-1.57	1.57	-1.57	1.57	-1.57	1.57

Table 3. The limitations of joint rotations for CMA-ES. The unit of the numbers is radian.

Stage	data	latent prior	shape prior	kinetic prior	interaction term
Stage1	1.0	4040.0	100.0	1000.0	0.0
Stage2	1.0	404.0	50.0	500.0	0.0
Stage3	1.0	57.4	10.0	250.0	0.0
Stage4	1.0	1.78	5.0	200.0	4500.0

Table 4. The loss weights of kinematic optimization in each stage.

state as the same as the state from linear interpolation. In the training phase, we add random noises in the character state to simulate the discrepancy of real simulation. The distribution prior is trained on a single NVIDIA TITAN RTX GPU with a learning rate of 0.0001 and a batch size of 32.

1.4. Sampling details

We describe implementation details of the neural motion control. The character state $\boldsymbol{s}$ consists of pose $\boldsymbol{q}$ and velocity $\dot{\boldsymbol{q}}$. The first 6 dimensions of pose are global translation and global rotation. The rest are joint rotations that are represented by axis-angle. Besides, $\dot{\boldsymbol{q}}$ contains 3-dimension base linear velocity, 3-dimension base angular velocity and 51-dimension joint angular velocity. The total dimension of character state is 114. In the physics simulator, the rotations are represented by quaternion.

We do not directly control the root joint, thus the target pose has the same dimension as the DOF (degree-of-freedom) of moveable joints, which is 51. Given a target pose, we first compute torques from the PD controller and limit the torques in a reasonable range. The parameters are shown in Tab. 1. Finally, the torques are applied to the character via torque control mode. We simulate 8 times for a given target pose and re-calculate the torques based on the target pose and simulated pose in each time. When applying the distribution prior in neural motion control, we use the first frame of reference motion to initialize the physical character.

1.5. Optimization details

Our optimization has four stages. The only difference among each stage is the loss weights for each term. The different loss weights promote the optimized results from coarse to fine. As shown in Tab. 4, the optimization with the weights in the first three stages can obtain proximate results. We only apply the interaction constraint in the last stage to get accurate ground contact.

1.6. Details on SDF

The SDF representation is similar to [4], in which the scene is used to constrain a single human pose. We use a uniform voxel grid with the size $256\times256\times256$ to represent the field. The trilinear interpolation is used for the discretization of the 3D distance field with the limited grid resolution. The resolution is enough to obtain coarse contact for physics-based motion capture with reasonable computational complexity and memory consumption.

2. Success rate

Since our method is based on sampling, the reconstructed control is not guaranteed to be successful after a single run of the sampling algorithm [6]. The character may fall and be unable to finish the complete motion. Thus, the success rate is an important metric to evaluate our method. For sampling-based motion control, the sampling distribution is the most important influencing factor for the metric of success rate. CMA-ES-based method [5] learn from previous trials to update the distribution via online adaptation and might require more trials for motion capture tasks. The initial several trials draw samples randomly and blindly and are very likely to fail, which results in a low overall success rate. The limitation is also demonstrated in a recent work [11]. With the well-trained prior, our method samples from the regressed distribution and has a high success rate for all trials. In Tab. 5, we follow [11] and [6] to conduct a comparison on the lift leg motion. The success rate is 97% and 90% for our method and [5], but [6] is 83%. The comparisons illustrate the advantage of our approach from a different perspective. In addition, to increase the number

Method	Liu *et al*. [6]	Liu *et al*. [5]	Neural MoCon
Success rate	83%	90%	97%

Table 5. The comparison on the lift leg motion with the metric of success rate.

of samples and saved samples at each iteration can improve the success rate. Furthermore, when the tracking is fail, we can also run multiple times on the same problem to allow a user to explore different possible reconstructions.

3. More results and discussions

We show more qualitative results in this section to demonstrate the performance of our method. In Fig. 2, the results on GPA, 3DOH and Human3.6M dataset show that our method is robust to complex terrains, occlusions and body shape variations. Furthermore, we apply the estimated pose from neural motion control to SMPL model and get the skinning mesh. It shows the obtained meshes are natural and accurate. Since the collision detection is conducted on the primitives of physical character, the shape discrepancies of hand and foot cause a slight interpenetration on the skinning mesh. We will detect mesh-level collision or design more delicate characters to prevent these artifacts in the future work.

4. Video

In the video, we show the qualitative comparisons with VIBE, DMMR, and PhysCap. Due to the hard physical constraints, our method can prevent floor interpenetration. Most of the foot sliding is also avoided by applying lateral friction. To demonstrate the performance of our method on complex terrain, we conducted a comparison with PhysCap on the GPA dataset. We used the original character model of PhysCap. Although PhysCap can obtain smooth motion, the wrong contact states for uneven terrain scenario result in floating motion. However, since the interaction constraint is used in kinematic optimization, our method can produce a physically plausible and high-quality motion with the proposed neural motion control.

5. Why neural motion control?

To build a 3D human dataset with accurate force annotations is complex and expensive [12]. The joint torques cannot be measured non-intrusively and therefore need to be derived using computationally expensive optimization techniques. Furthermore, the torques for different subjects with different body shapes have large variances. It results in a poor generalization ability for the network that directly regresses joint torques. However, the distribution prior is a network that regresses the target pose distribution. With the dense supervision from pseudo ground truth and the two-branch decoder, the network is easy to be convergent. In addition, with the sampling, the neural motion control is more general to complex terrain, body shape variations, and diverse behaviors.

Compared to CMA-ES based method, the existing sampling-based motion control first relies on CMA-ES to adapt the distribution via evaluating plenty of samples, which is time-consuming. Our network-based prior avoids such distribution adaptation, and elite samples can be directly obtained from the regressed distribution. Our method saves a lot of sample evaluations compared to [5]. Furthermore, the distribution adaptation relies on random samples from an initial distribution to update the distribution via CMA-ES, which imposes uncertainty for the motion capture. The proposed prior avoids the uncertainty, and the precise control can be acquired by sampling from the distribution of network output, which is the same as CMA-ES-based approaches. Combing neural networks and sampling-based motion control provides a feasible solution to achieve real-time physics-based motion capture though there is still a gap for this goal.

References

[1] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021. 1

[2] Nikolaus Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation*, pages 75–102, 2006. 1

[3] Nikolaus Hansen. The CMA evolution strategy: A tutorial. *CoRR*, abs/1604.00772, 2016. 1

[4] Mohamed Hassan, Vasileios Choutas, Dimitrios Tzionas, and Michael J Black. Resolving 3d human pose ambiguities with 3d scene constraints. In *ICCV*, 2019. 3

[5] Libin Liu, KangKang Yin, and Baining Guo. Improving sampling-based motion control. In *Comput. Graph. Forum*, volume 34, pages 415–423, 2015. 3, 4

[6] Libin Liu, KangKang Yin, Michiel van de Panne, Tianjia Shao, and Weiwei Xu. Sampling-based contact-rich motion control. In *SIGGRAPH*, 2010. 3, 4

[7] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015. 1

[8] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 1

[9] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013. 1

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 32:8026–8037, 2019. 1

[11] Kaixiang Xie and Paul G Kry. Inverse dynamics filtering for sampling-based motion control. In *CGF*, 2021. 3

[12] Petrissa Zell, Bodo Rosenhahn, and Bastian Wandt. Weakly-supervised learning of human dynamics. In *European Conference on Computer Vision*, pages 68–84. Springer, 2020. 4